Java – classes e objetos

Prof. Mauricio Prado Catharino

Problemas com a Programação Procedural

- Mudanças de requisitos;
- Mudança de desenvolvedor;
- Muitas pessoas responsáveis por colocarem o mesmo código em vários lugares

Problemas com a Programação Procedural

- Mudança de desenvolvedor.
  - Necessidade de ler o código que foi escrito por outro desenvolvedor, e descobrir como ele funciona, isso se torna impossível em sistemas de grande porte.

Problemas com a Programação Procedural

- O problema do paradigma procedural é que não existe um forma simples de criar conexão entre os dados e funcionalidades.

- No paradigma orientado a objetos é muito fácil ter essa conexão através de recursos da linguagem.

Problemas com a Programação Procedural

- Exemplo clássico da validação CPF.
  - Validar o CPF em todos os formulários do programa.

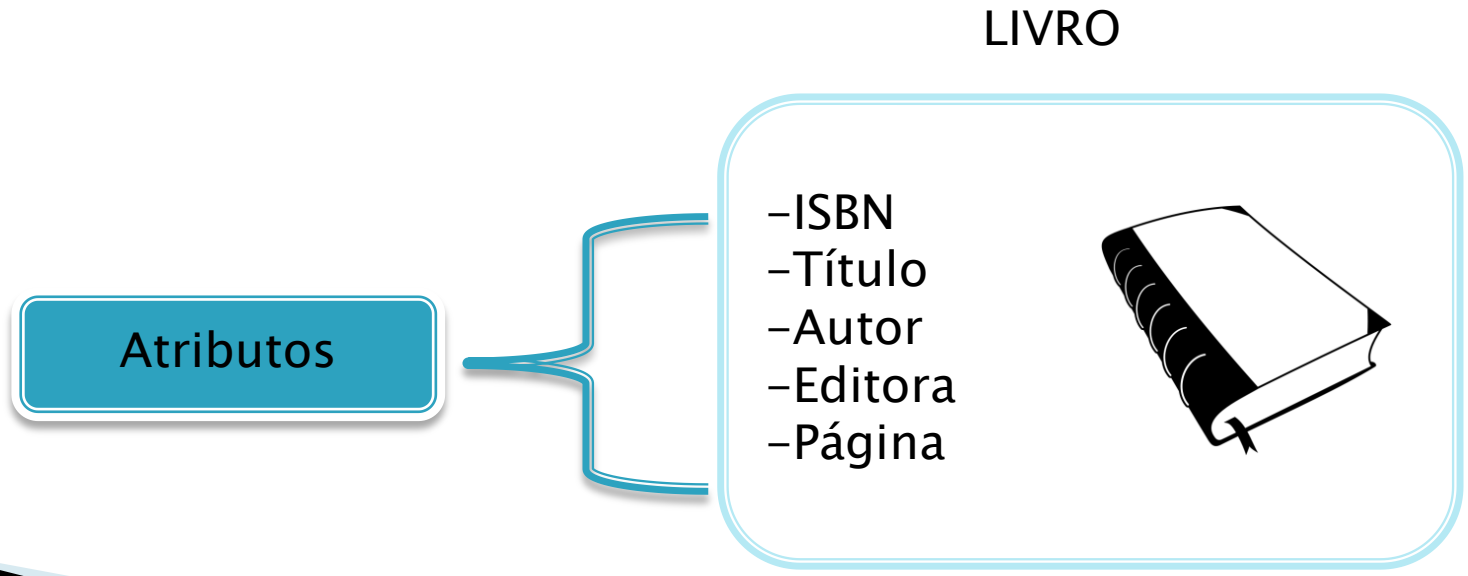
- Problemas como:
  - Rescrita e código;
  - Manutenção do código;
  - Possibilidade de erro em um dos formulários;
  - Custo de programação.

Orientação a objeto

- Benefícios:
  - Escrever menos códigos;
  - Concentrar responsabilidades nos locais certos;
  - Flexibilização da aplicação;
  - Encapsular lógica de negócio.
  - Polimorfismo (variação de comportamento)

Classe

- Uma classe representa um tipo de dados.
- Uma classe é uma estrutura definida pelo programador.



Classe

- A palavra classe vem da taxonomia da biologia. Todos os seres vivos de uma mesma classe biológica têm uma série de **atributos** e **comportamentos** em comum, porem não são iguais.
- Por exemplo o homem é da classe dos mamíferos.

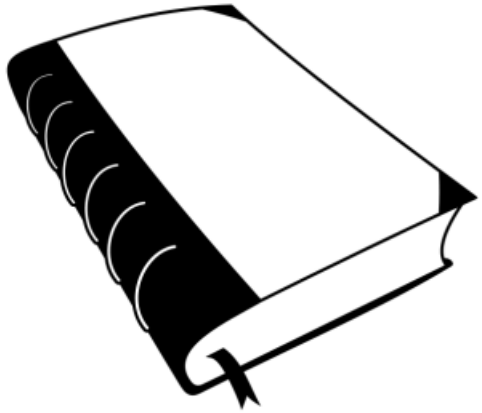
Classes e seus métodos

Emprestar

Devolver

Reservar

Obter Autor



Métodos

Atributos x Métodos

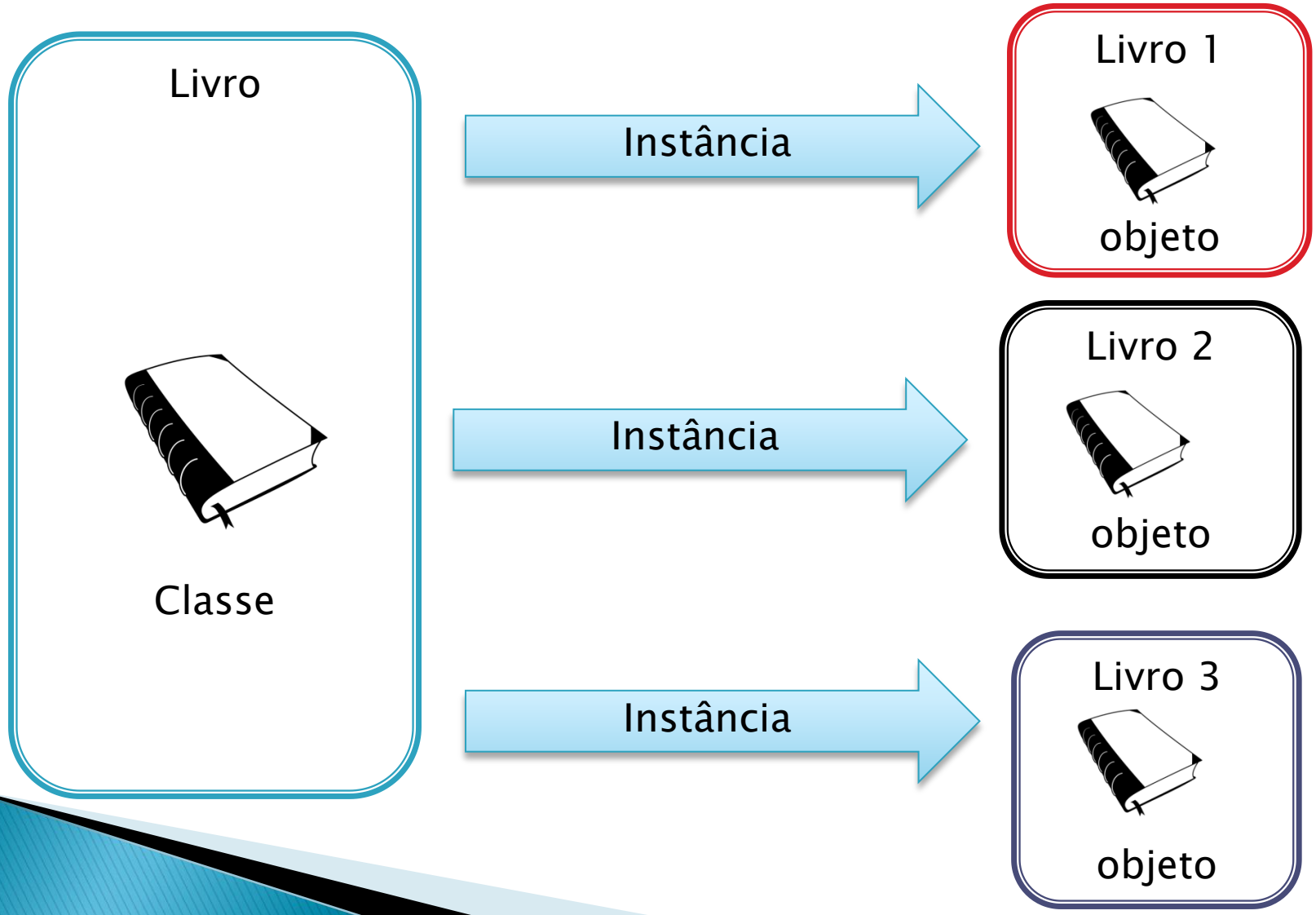
- Atributos
  - Característica da classe.
  - Representados por substantivos.

- Métodos
  - Operações que as classes são capazes de realizar.
  - Representados por verbos.

Classes x Objetos

- Classes são estruturas utilizadas para construir os objetos que são instanciados por estas.

Classes x Objetos



Classes x Objetos

- Um outro exemplo bacana é uma receita de bolo. A pergunta é: você come um receita de bolo?
- Não, mas podemos instanciá-la, criar um bolo a partir das especificações dessa classe.
- Podemos criar a partir dessa receita centena de bolos, semelhantes ou mesmo idênticos, porem são **objetos** diferentes

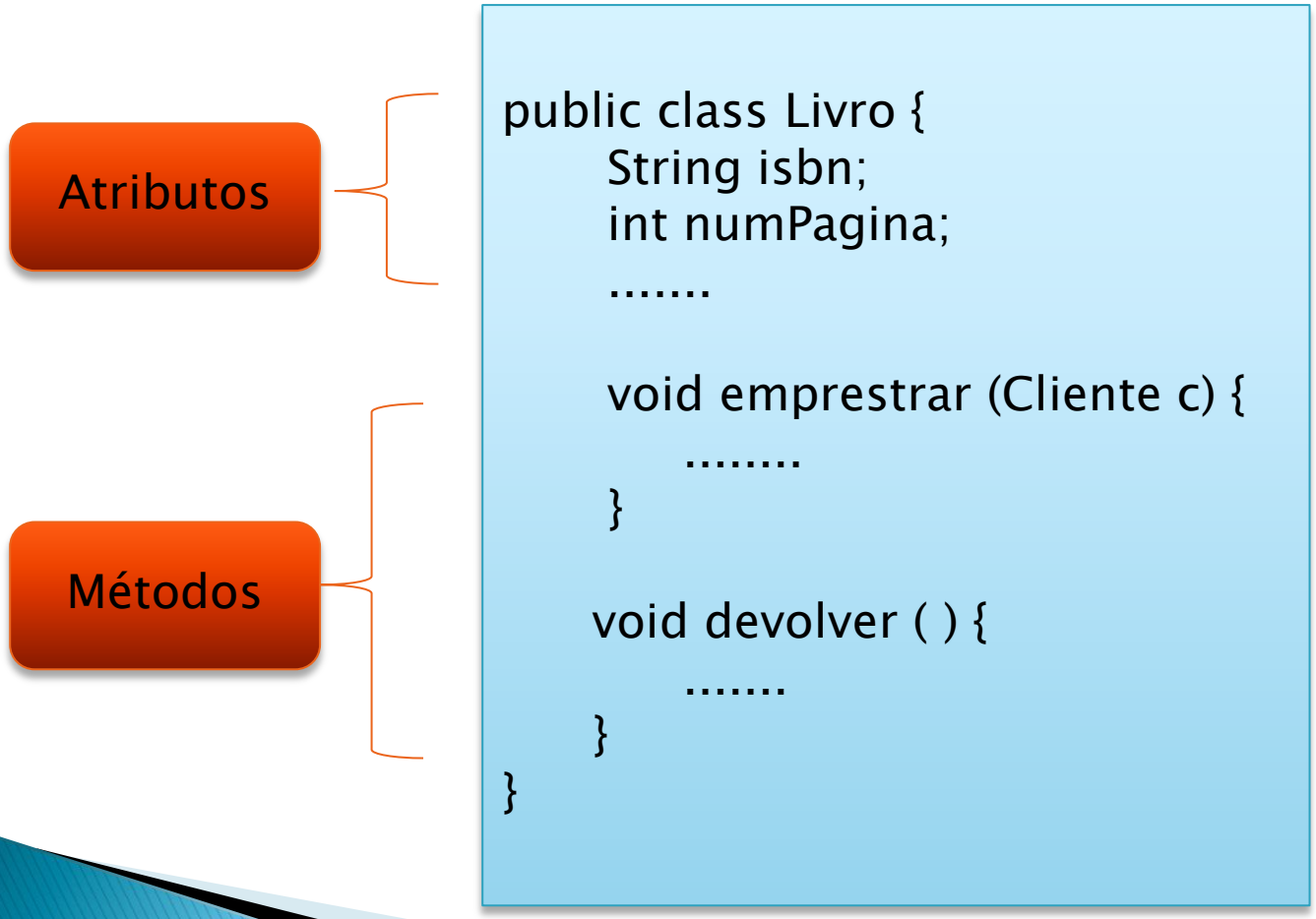
Declarando classe no Java

- No Java as classes são declaradas utilizando a palavra ***class.***

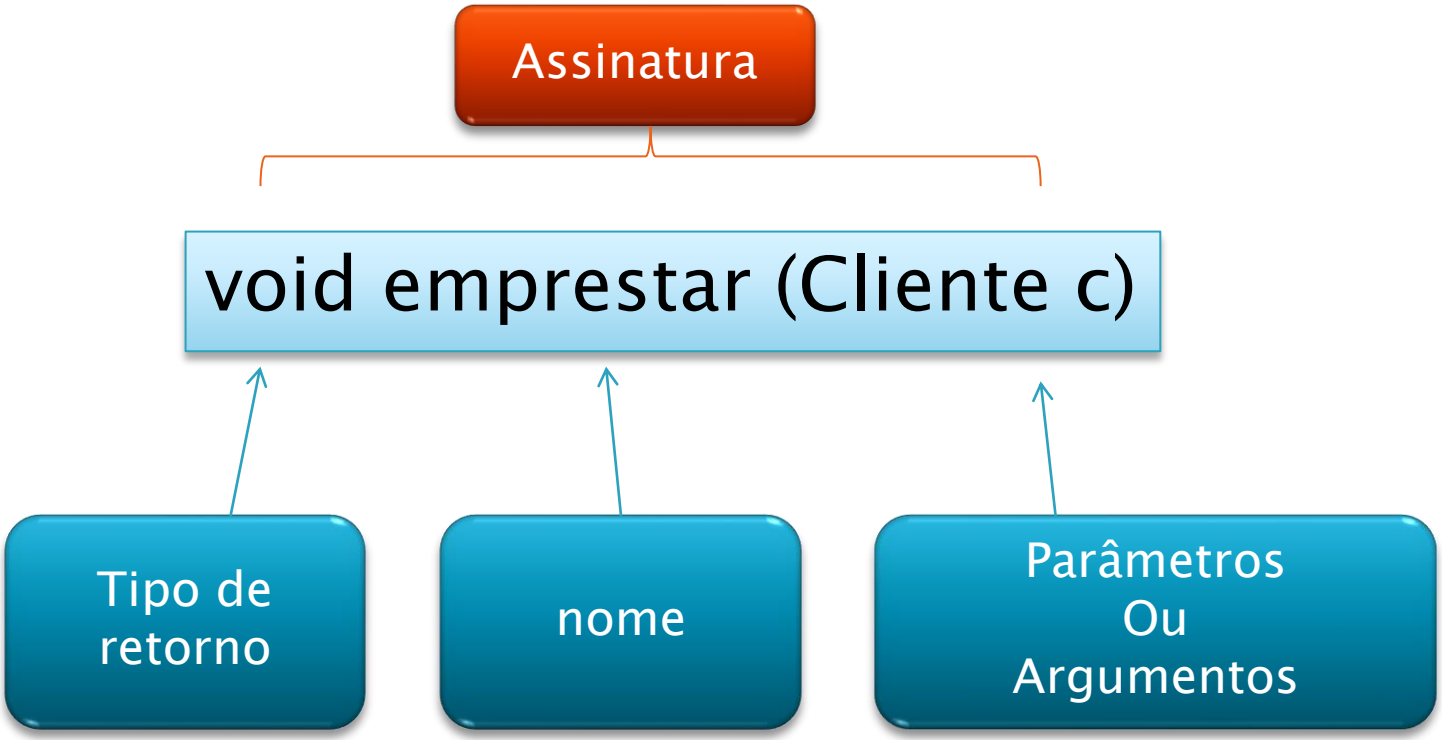
```
public class livro   {
        ..............
}
```

- Uma arquivo .java pode ter apenas uma classe declarada como pública dentro dele.

Declarando métodos e atributos



Assinatura de um método

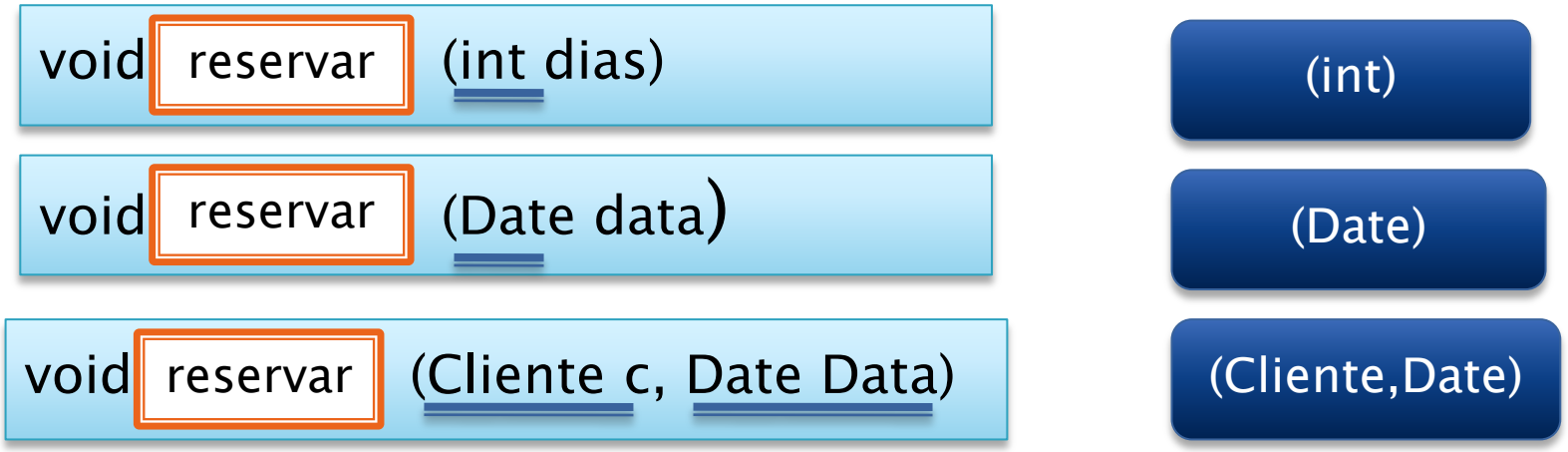


Assinatura de um método

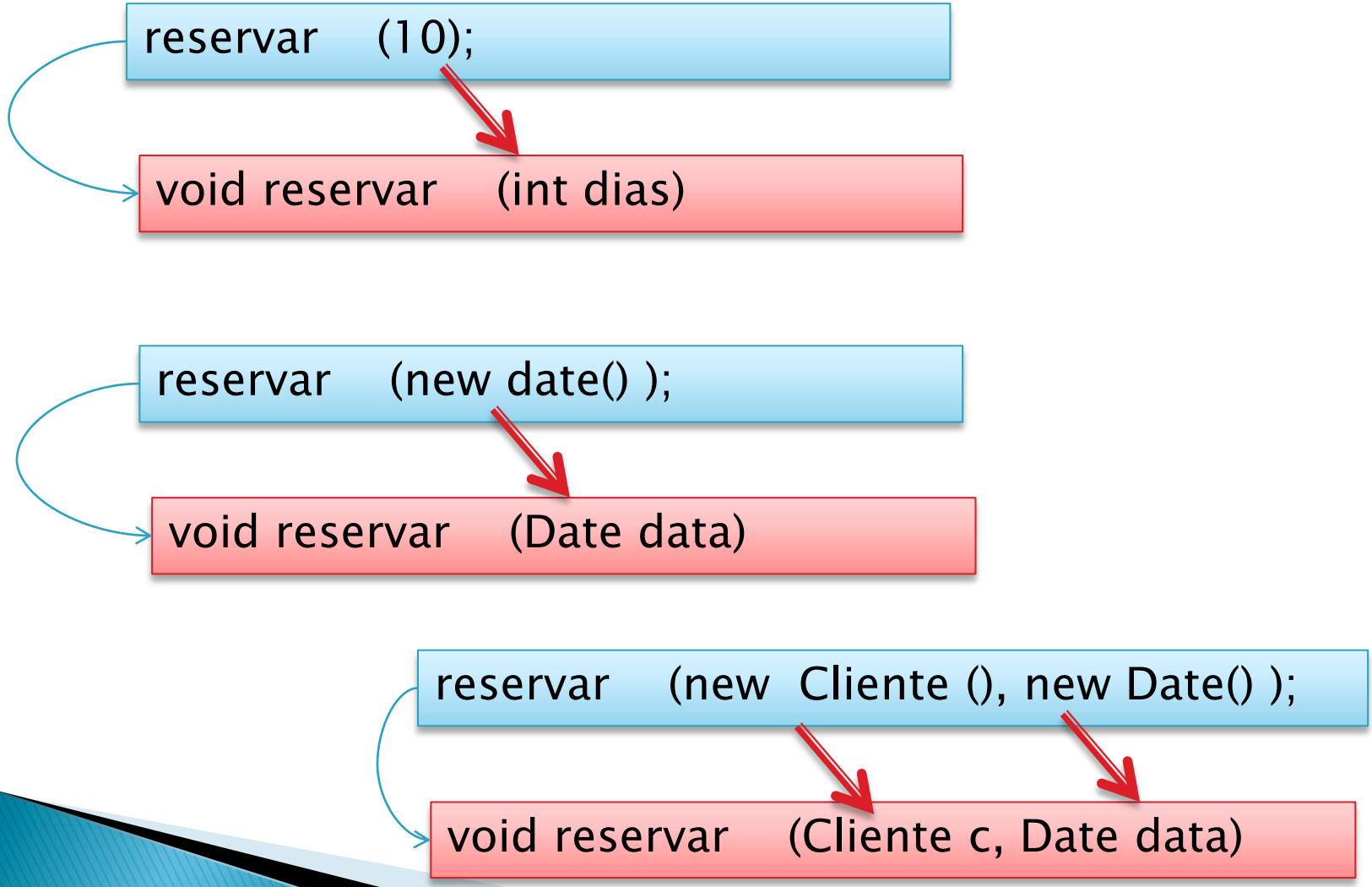
- Se o método não retorna valores, é utilizado o *void*.
- Um método pode ter zero ou mais parâmetros, e todo o parâmetro deve ter um tipo definido

Sobrecarga de métodos

- Sobrecarga de um método significa criar outros métodos com o mesmo nome, mas com assinaturas diferentes.



Sobrecarga de métodos



Criando e manipulando objetos

- Um objeto é sempre uma instância de uma classe.
- Para instanciar objetos é utilizado o new.

```
Livro livro1 = new Livro();
Cliente cliente1 = new Cliente();
```

- O objeto possui acesso ao que foi definido na sua estrutura (classe) através do “.”

```
Livro1. titulo = “Java”;
Livro1.emprestrar = (Cliente1);
```

Criando e manipulando objetos

- Cada objeto criado com o *new* é único.
- Os atributos de objetos diferentes pertencem apenas ao objeto.

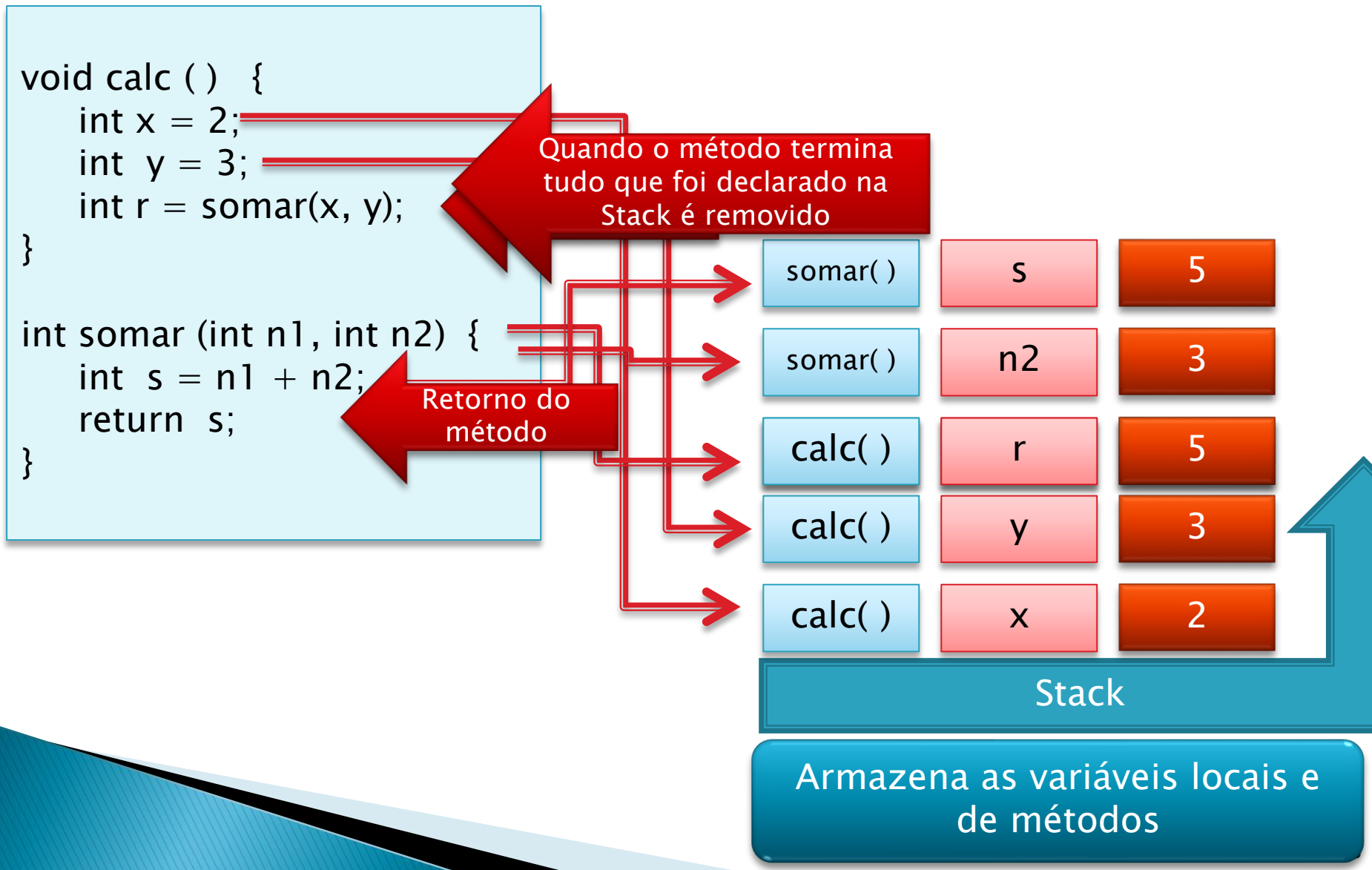
```
Livro livro1 = new Livro();
livro1.isbn = "1234";

Livro livro2 = new Livro();
livro2.isbn = "4321";


Livro livro3= new Livro();
livro3.isbn = "1212";
```

Objetos e referências

- Uma variável cujo o tipo é uma classe não guarda o objeto diretamente.
- A variável guarda uma referência ao objeto.
- O *new* aloca um área de memória e retorna a referência da área de memória alocada.
- As variáveis declaradas em métodos são criadas em uma área de memória conhecida como *stack*.
- Os objetos são criados em uma área de memória conhecida com *heap*.

Objetos e referências – Stack

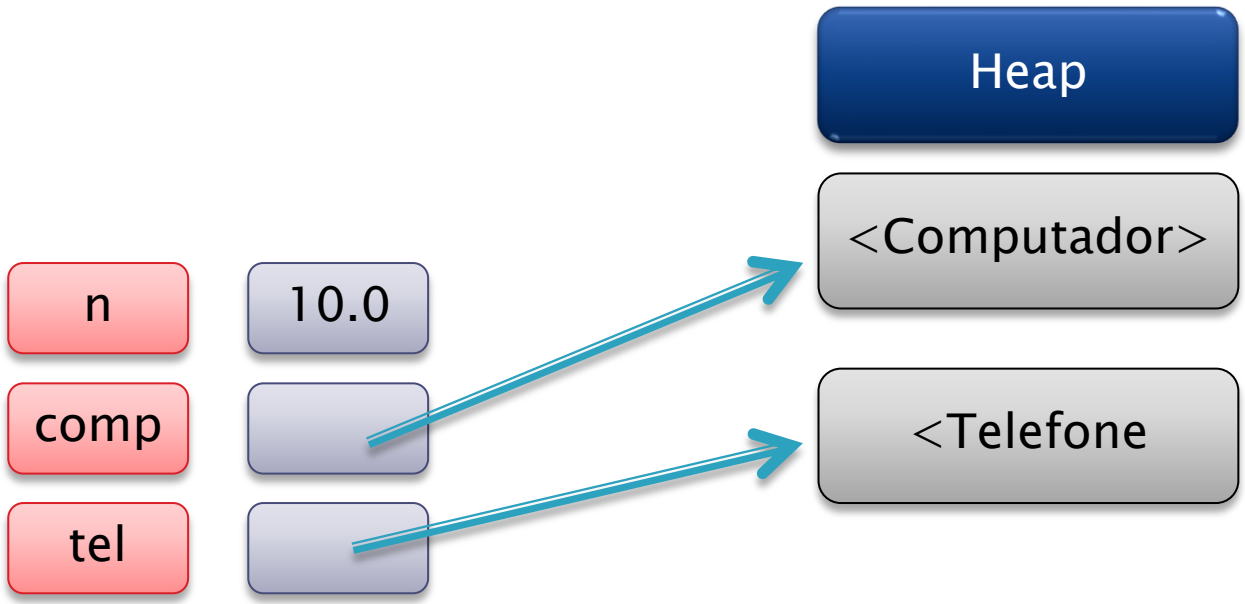
```
void calc ( )  {
    int x = 2;
    int  y = 3;
    int r = somar(x, y);
}

int somar (int n1, int n2)  {
    int  s = n1 + n2;
    return  s;
}
```



Objetos e referências – Heap

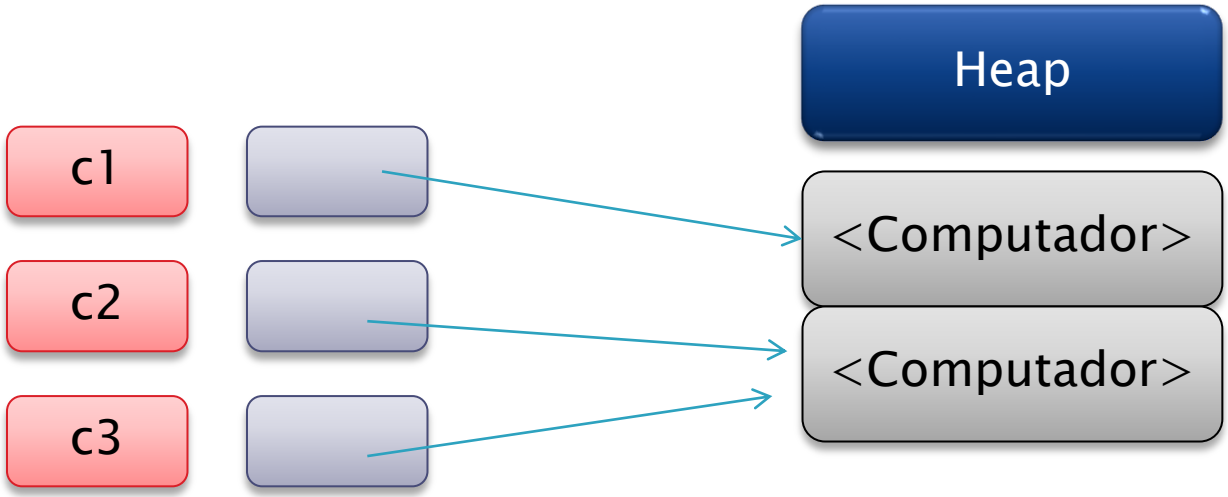
```
double n = 10.0;
Computador comp = new Computador();
Telefone tel = new Telefone();
```

O operador new cria um objeto no Heap



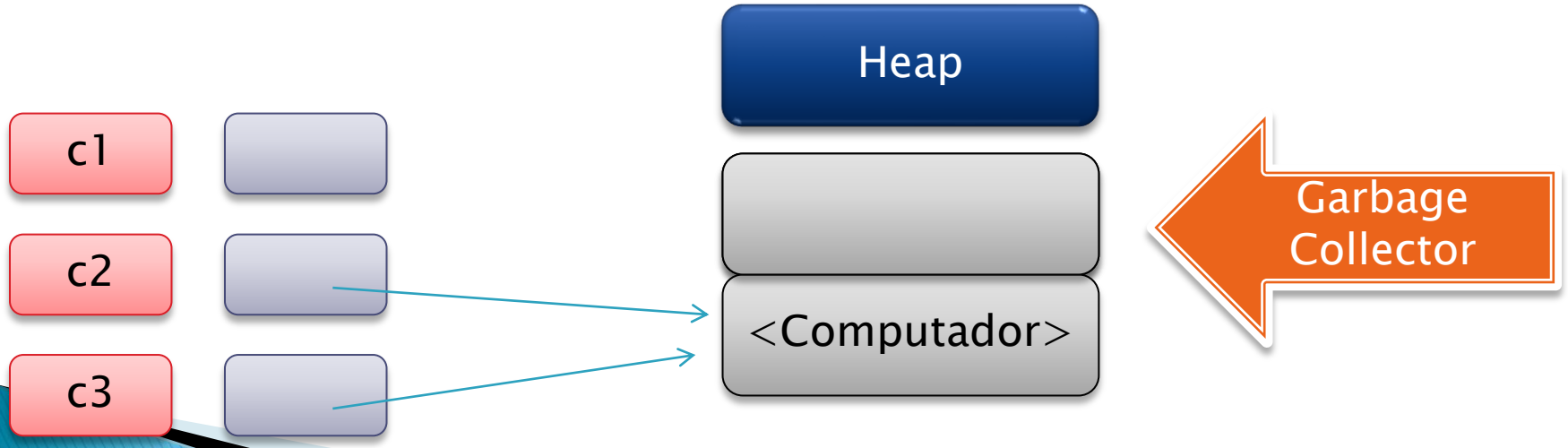
Objetos e referências – Heap

```
Computador c1 = new Computador();
Computador c2 = new Computador();
Computador c3 = c2;
c1 = null;
```



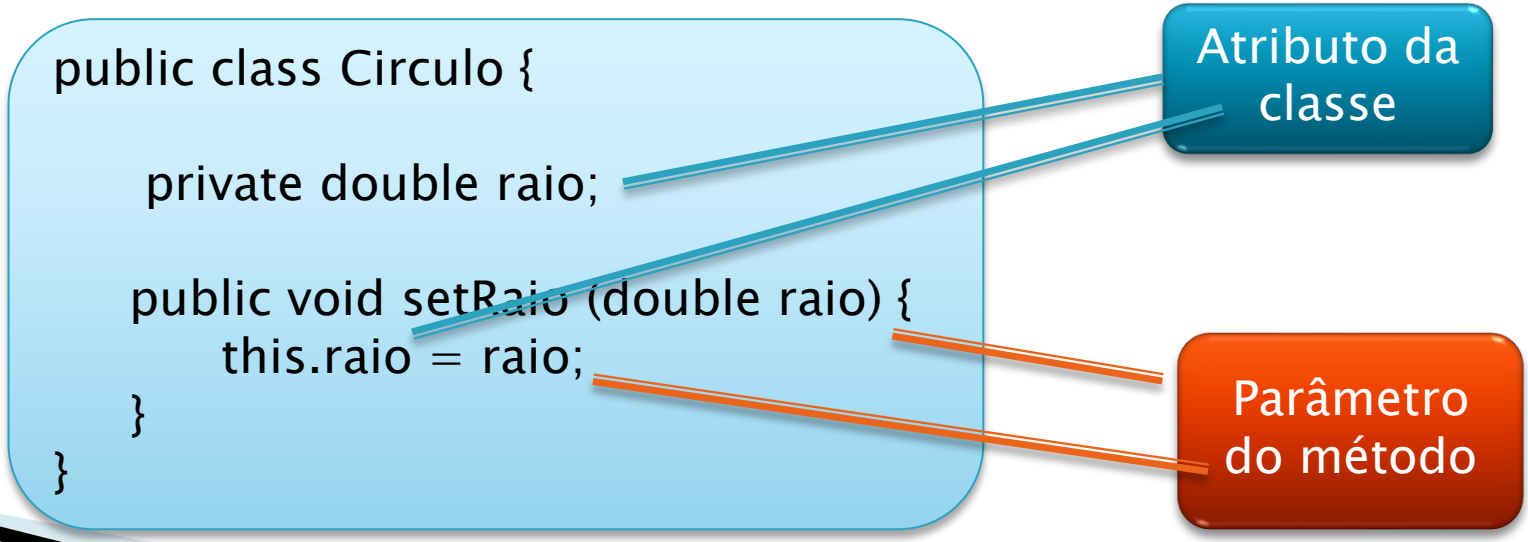
Garbage Collector

- Serviço da JVM que é executado em segundo plano.
- Procura objetos no Heap que não são mais utilizados pela aplicação e os remove.
- Não pode ser controlado pelo desenvolvedor.



Operador this

- Usado basicamente em duas situações:
  - Diferenciar um atributo do objeto de um argumento do método.
  - Fornecer a referência do próprio objeto para outro método.




Dúvidas?

Exemplo de Classes

```java
public class Aplicacao {
  public static void main(String[] args) {

      Pessoa p1 = new Pessoa();
      p1.nome = "José";

      Pessoa p2 = new Pessoa();
      p2.nome = "Maria";

      Pessoa p3 = new Pessoa();
      p3.nome = "Carlos";

      p1.receber(5);
      p1.receber(7);

      p1.dar(2, p2);

      p3.receber(20);
      p2.dar(4, p3);

      System.out.println(p1.nome + " ==> " + p1.numFigurinhas);
      System.out.println(p2.nome + " ==> " + p2.numFigurinhas);
      System.out.println(p3.nome + " ==> " + p3.numFigurinhas);
  }
```

```java
public class Pessoa {

    String nome;
    int numFigurinhas;

    void receber (int numFigurinhas){
        this.numFigurinhas += numFigurinhas;
    }

    void dar (int numFigurinhas, Pessoa p){
        this.numFigurinhas -= numFigurinhas;
        p.numFigurinhas += numFigurinhas;

    }
```

Exemplo de Referência a Objeto

```
public class TestaReferencia {
    public static void main(String args[]){

        Conta c1 = new Conta();
        c1.deposita(100);

        Conta c2 = c1;  // c2 recebe a referencia de c1
        c2.deposita(200);

        System.out.println(c1.saldo);
        System.out.println(c2.saldo);

    }
}
```

```
public class Conta{
    double saldo    ;
    int numConta;

    void saca(double quantidade){
        double novoSaldo = this.saldo-quantidade;
        this.saldo = novoSaldo;
    }

    void deposita(double quantidade){
        this.saldo += quantidade;
    }

}
```

Exercício

1. Crie um programa em java que acesse uma classe conta que possua os métodos sacar e depositar, o programa deverá passar o nome do cliente o valor do deposito e o valor do saque e apresentar o nome, o número da conta e o saldo do referido cliente.
2. Em relação a sobrecarga de métodos crie um exemplo de uma classe que apresente esta característica.